# QGrain

QGrain is an easy to use software that can unmix the multi-modal grain size distribution to some single modals.

It's written by Python. This makes it can benefit from the great open source and scientific computation communities.

QGrain is still during the rapid development stage, its functionalities and usages may changes many and many times. And of course, there probably are some bugs. We are very sorry for its immaturity.

We are really hope to receive your feedbacks. Whatever it's bug report, request of new feature, disscusion on algorithms.

Moreover, we are looking forward that there are some partners to join the development of QGrain.

If you have any idea, you can contact the authors below.

## Authors


- Yuming Liu

  liuyuming@ieecas.cn

# How to install QGrain

**QGrain** is written by Python. So, it's **cross-platform**.

This means it can be installed on many kinds of systems, like **Windows**, **Linux**, **Mac OS X**, etc.

Moreover, due to Python's features, there are two ways to install QGrain.

## 1. Use executable file

We have packed the executable setup file for **Windows** users. Because Windows does not have Python internally installed, and Windows users are not used to the command-line interface.

You can download the latest version of QGrain from here.

When the setup file was downloaded, what you need to do is double-clicking the setup file to execute the installation program.

All things are similar to other softwares on **Windows**. And then, you can see the shortcut on the desktop.

## 2. Use Python

This method is for the user who has experience in Python and Shell (i.e. command-line interface). It uses the Python interpreter to execute the source codes of QGrain.

The advantage of this method is its expansibility. Through the `pip` (the package installer for Python), you can easily download and update QGrain. If you are not satisfied with some functionalities of QGrain, or there are some small bugs, you can modify the codes by yourself to solve these problems instead of waiting for a new update to fix them.

1. Install Python

   For Linux and Mac OS X users, you may have the built-in Python3 interpreter. You can run the command `python` or `python3` in your terminal to check if Python is existing.

   Note: Using `python` or `python3` depends on the alias or the filename of your Python3 interpreter, rather than you can choose Python2 or Python3 at liberty. Python2 is too old and has been obsoleted.interpreter

   If you have Python3 installed on your computer, you can see the text like below (test on Ubuntu 18.04 LTS). You can see the version of your Python. And the `>>>` symbols hint you that you have entered the interactive model of Python's interpreter. Type `quit()` to quit this model and back to the terminal.

   ```
   your_user_name:~$ python3
   Python 3.6.9 (default, Nov  7 2019, 10:44:02)
   [GCC 8.3.0] on linux
   Type "help", "copyright", "credits" or "license" for more information.
   >>>
   ```

If you have not Python3 yet, you can visit the official website of Python to download and install it.

Note: If you are a **Windows** user, remember to check the **Add Python to Windows PATH** option while installing Python. If not, some commands in this tutorial may be **invalid**, due to the `PATH` mechanism of Windows.

2. Get source codes and install QGrain

   We have upload QGrain to PyPI. So, you can get the codes very expediently through `pip`. By running the following command, you can get QGrain module installed.

   ```
   pip install QGrain
   ```

   For Chinese users, this command below may be faster.

   ```
   pip install -i https://pypi.tuna.tsinghua.edu.cn/simple QGrain
   ```

   Note: On some systems, the name of `pip` of Python3 may be `pip3`.

   Run `pip install -U QGrain` to check and update QGrain.

   In addition, you can download the codes from Github. The link of our repository is over here.

   - You can clone the repository by running the following command if you have `git` installed.

     ```
     git clone https://github.com/QGrain-Organization/QGrain.git
     ```

   - Or, you can download the pure codes directly by clicking here.

     If you choose to download the zip file, please extract it.

   Then, you should change the directory to QGrain's folder, and run the following command.

   `python setup.py install` or `pip install .`

3. Run QGrain

   Now, everything is ok.

   Then you can run `qgrain` command in your terminal.

   If it goes well, you can see the app interface below.

   If the command `qgrain` was not found. It may be caused by the lack of Python's directory in the Windows `PATH` environment variable.

   Because QGrain will generate an executable file (e.g. qgrain.exe on Windows) to start this app, it's located at the `Scripts` sub-folder of your Python. If this folder is not in your `PATH`, the system can not find this executable file, and the command will raise the not found error.

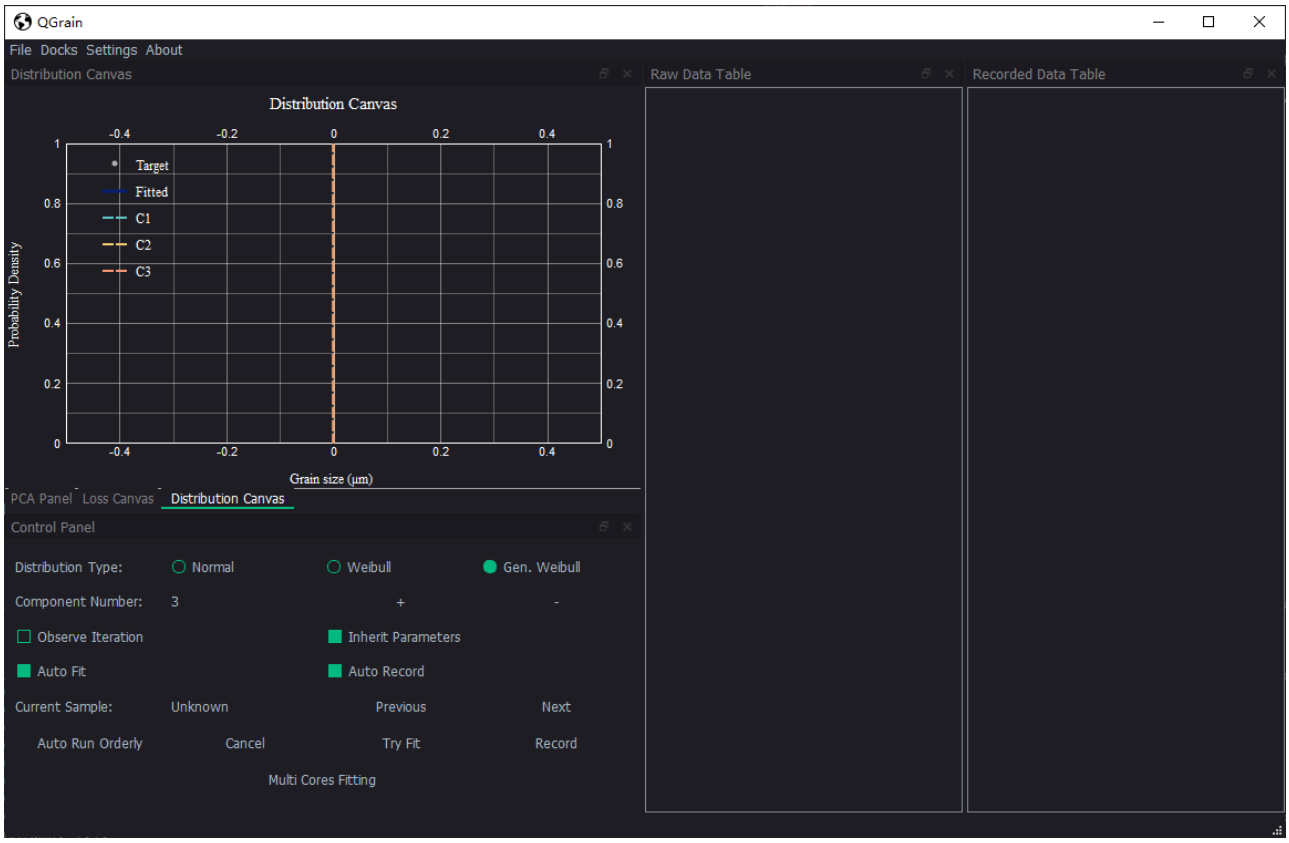
   In order to let the modification of `PATH` come into force, you need to restart your PC.

   If restart did not solve your problem, you may need to check the `PATH` variable and **APPEND** the root and `Scripts` folders of Python to the `PATH`.

For some users who do not want to modify the `PATH`, there also is another way to start QGrain.

1. Type `python` or `python3` to enter your Python interpreter.
2. Type following codes.

```
import QGrain
QGrain.main()
```



# Overview of UI

## Layout

QGrain consists of several docks, i.e. small child windows. These docks may contain several widgets that collaborate to provide a certain function for users.

The major feature of docks is its flexibility, that is, they are dockable, floatable, movable, scalable, divisible, and closable. These features make you can adjust the layout as you please if you are not satisfied with it. The customized layout will be stored after you closed this app.

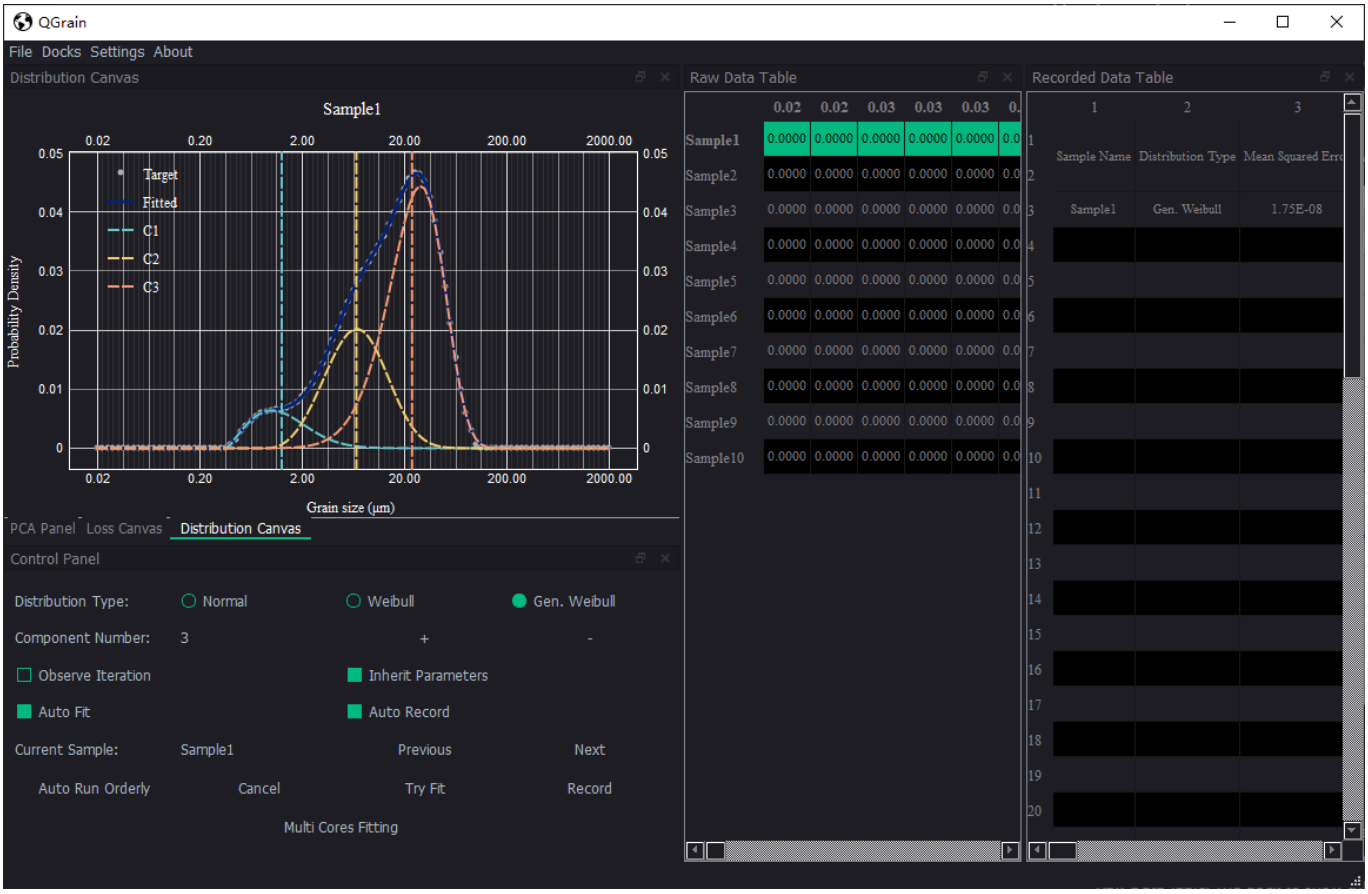
Each dock can dock in the main window and become the child window of the main winodw. Each child window has its own title bar that displays its name. By double-clicking its title bar or clicking its *Reset* button, you can float this dock, and it will become an independent window. Also, you can drag the title bar to make it floating.

When you dragging this dock, there is a highlight rectangle to show you the area it will be put in. By this method, you can make the floating dock become docked again. Or, you can move it to another place. If you drag it around another dock, it will split and share the space with that dock. If you drag it on that dock, it will be tabbed, which means there is only one dock that can ben displayed at the same time, and you can click the tabs below to switch between them.

If you want to change the sizes of docks, you can drag the separator (i.e. the boundaries of docks) to adjust it. By clicking the *Close* button of the dock, you can close the needless dock. If you want to display a dock that has been closed before, you can click the **Docks** menu and select the corresponding option to realize it again.

By default, QGrain puts the docks which have a canvas (i.e. chart) in the top left corner, and all these docks are tabified to reduce the occupancy of space. In order to control this app conveniently, the dock of the `Control Panel` is in the bottom left corner alone. On the right, there are the docks to display the raw data and recorded results.

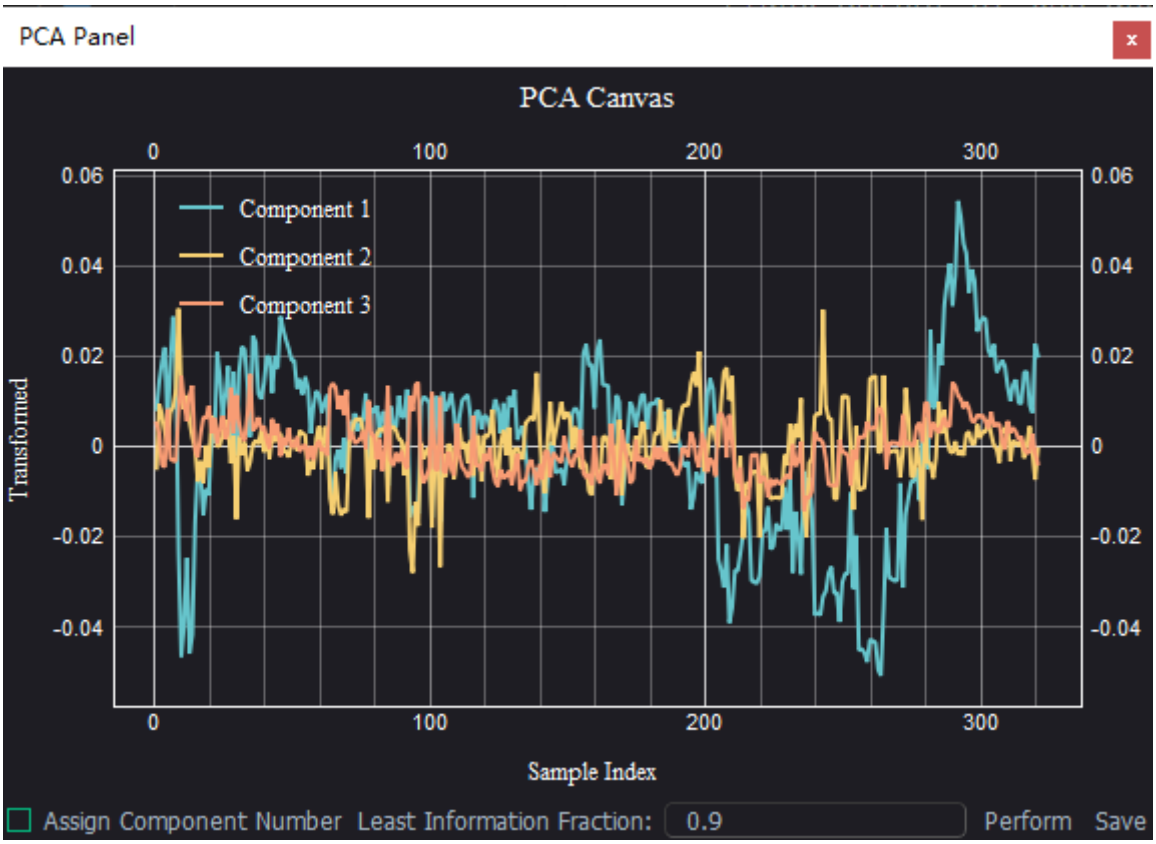
If you want to reset the layout to default, you can click the `Reset` option in the `Docks` menu.




# Docks

## PCA Panel

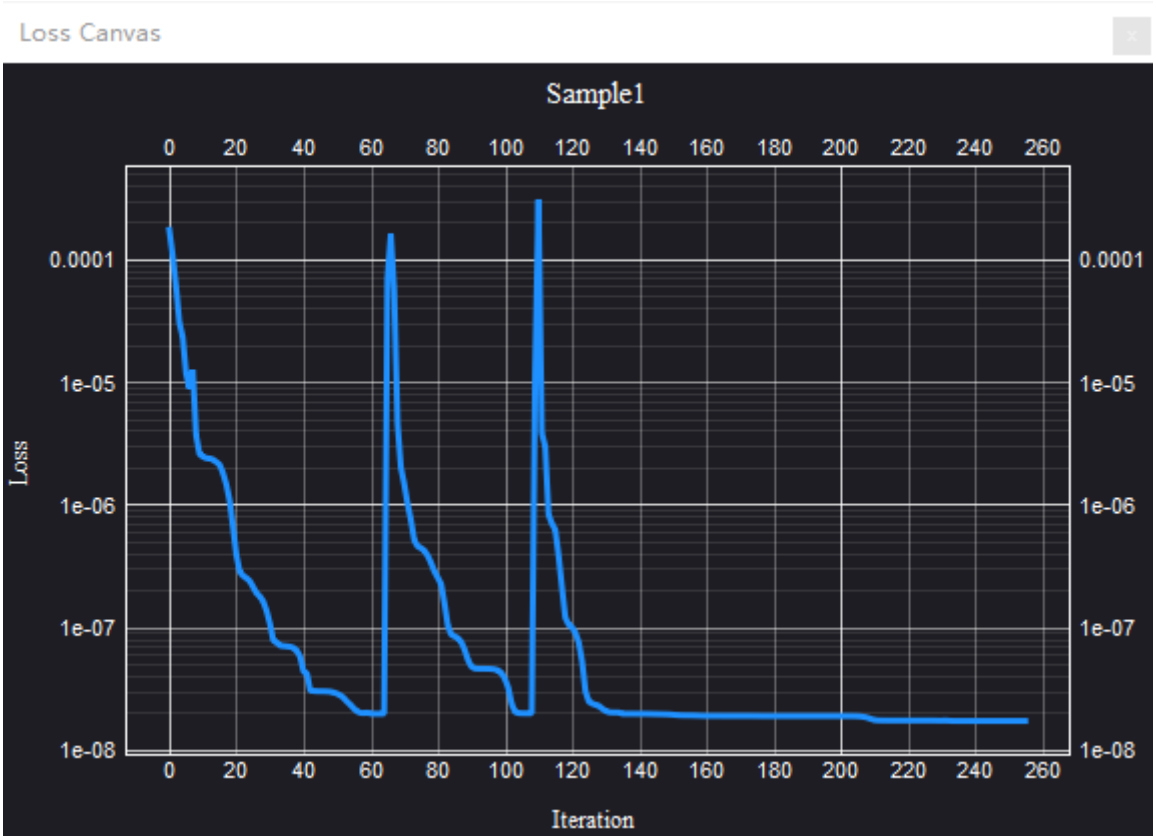
The dock to do principal component analysis (PCA) (see the wiki page for more details). It contains one canvas to show the result and some widgets bellow to control the algorithm.



## Loss Canvas

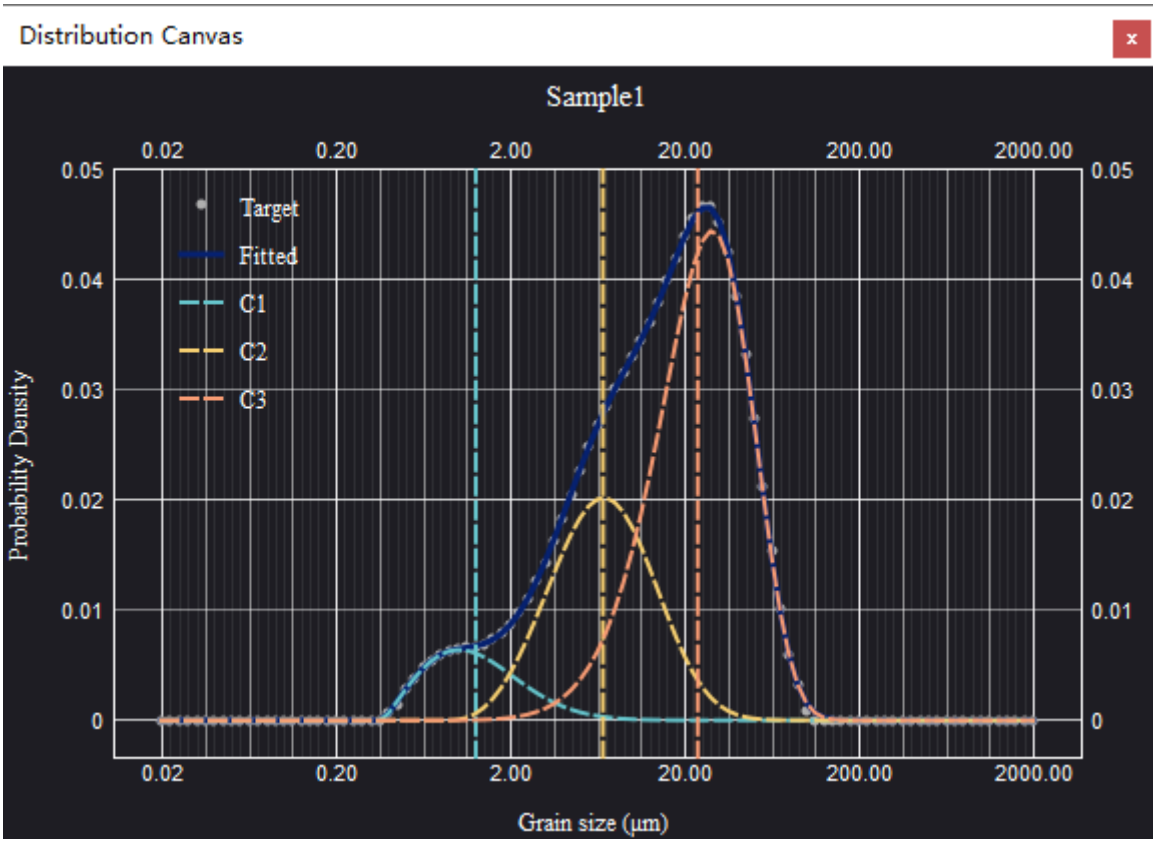
The dock to dynamically display the process of loss (i.e. the difference between observation and fitting result) changing.

Note: Only take effect when the **Observe Iteration** option of **Control Panel** is checked.




## Distribution Canvas

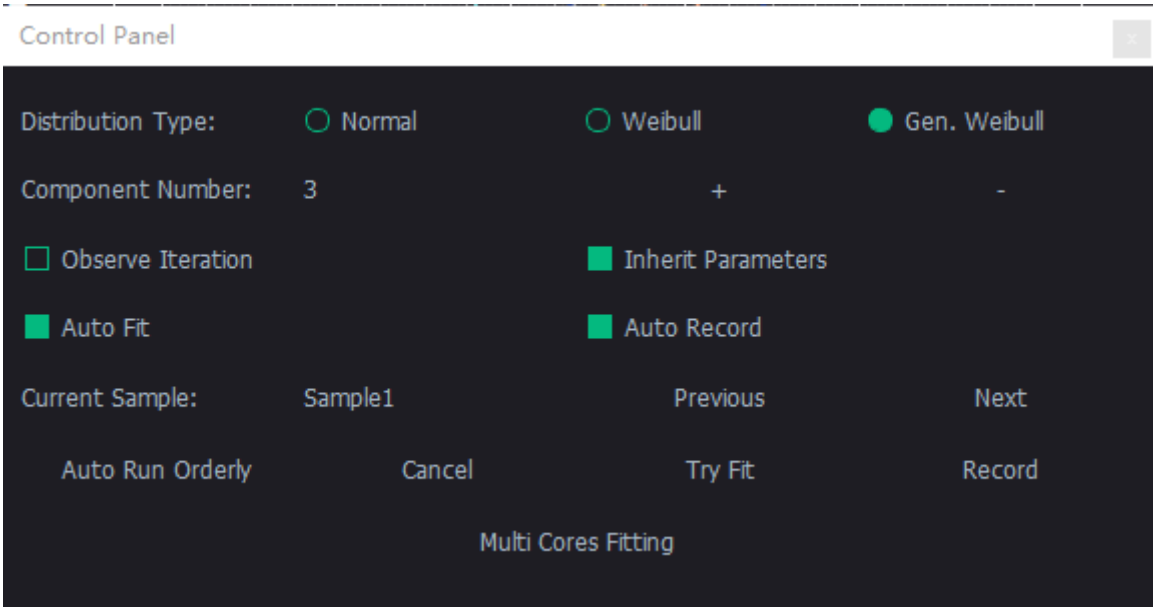
The dock to dynamically display the distribution of raw data and fitting result of current sample.




For the dock which contains canvas, you can follow the following mouse interaction to adjust the figure.

## Control Panel

The dock to control the fitting behaviours.




**Tips**

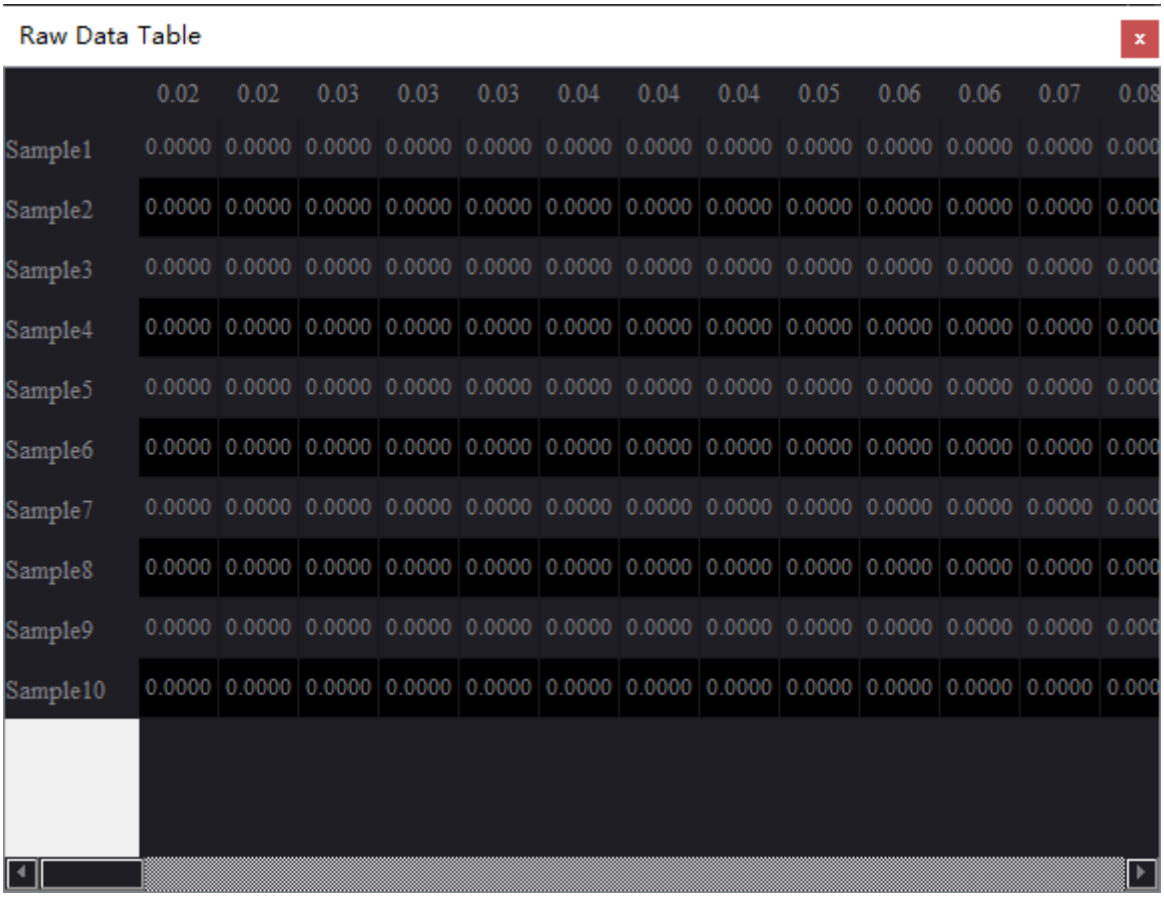
- Click the raido buttons of **Distribution Type** to switch the distribution function.
- Click the **+/-** button to add/reduce the component number you guess.
- **Observe Iteration**: Whether to display the iteration procedure.
- **Inherit Parameters**: Whether to inherit the parameters of last fitting. It will improve the accuracy and efficiency when the samples are continuous.

- **Auto Fit**: Whether to automaticlly fit after the sample data changed.
- **Auto Record**: Whether to automaticlly record the fitting result after fitting finished.
- Click the **Previous** button to back to the previous sample.
- Click the **Next** button to jump to the next sample.
- Click the **Auto Run Orderly** button to run the program automatically. The samples from current to the end will be processed one by one.
- Click the **Cancel** button to cancel the fitting progress.
- Click the **Try Fit** button to fit the current sample.
- Click the **Record** button to record the current fitting result.\nNote: It will record the LAST SUCCESS fitting result, NOT CURRENT SAMPLE.
- Click the **Multi Cores Fitting** button to fit all samples. It will utilize all cores of cpu to accelerate calculation.
- Move the lines in **Distribution Canvas** dock to set the expected mean values of each component, if it can not return a proper result and you make sure the component is correct.

## Raw Data Table

The dock to show the GSD data of samples.

By clicking the row of sample, you can set this sample as current sample and process it.




## Recorded Data Table

The dock to show the recorded fitting results.

Recorded Data Table

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | Component 1 | | | | | | | | | |
| 2 | Sample Name | Distribution Type | Mean Squared Error | Fraction | Mean (µm) | Median (µm) | Mode (µm) | Standard Deviation | Skewness | Kurtosis | Parameter 1 | Parameter 2 | Parameter 3 |
| 3 | Sample1 | Gen. Weibull | 1.75E-08 | 0.09 | 1.27 | 1.19 | 1.02 | 5.39 | 0.59 | 3.17 | 26.44 | 2.07 | 11.99 |
| 4 | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | |

# How to load grain size distribution data

In order to analyze the grain size distribution (GSD), the first necessary step is data loading. Click the **File** menu and select **Load** option to load GSD data from local files.

QGrain provides some built-in files of samples. Lookup the **samples** folder of QGrain to get them.

At present, QGrain supports three file formats, `.csv`, `.xls` (Excel 97-2003) and `.xlsx`.

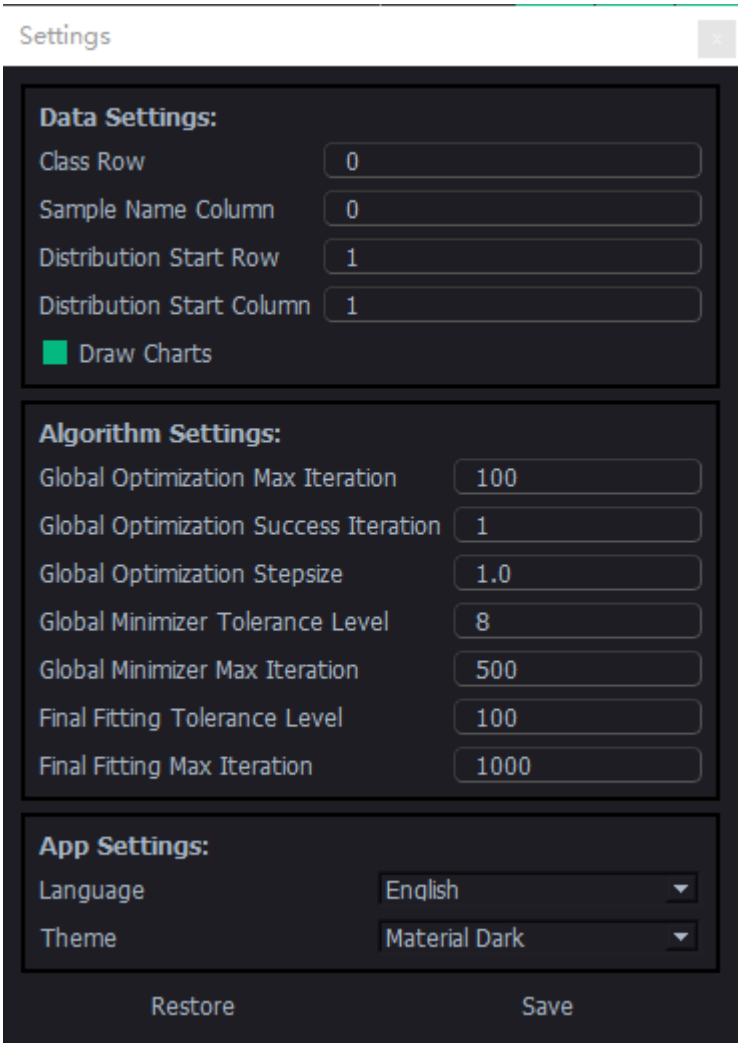
Note: For `.xls` and `.xlsx`, please put the data table at the **FIRST** sheet. Otherwise, the data can not be loaded.

By default, QGrain assume that the data layout follows:

- The first row should be the headers (i.e. the classes of grain size).
- The following rows should be the distributions of samples under the grain size classes.
- The first column shoud be the name (i.e. id) of samples.

If your layout of data file are not same as this, it will raise exceptions.

If you do not want to modify your data file, you can click the **Settings** menu to modify the default settings of loading data.



There are 4 parameters that control the data loader. They all are row/column index which start with 0 (i.e. 0 means the first row/column).

- Class Row: The row index of grain size classes.
- Sample Name Column: The column index of sample names.
- Distribution Start Row: The start row index (starts with 0) of distribution data. It should be greater than the row index of classes.
- Distribution Start Column: The start column index (starts with 0) of distribution data. It should be greater than the column index of sample name.

# The algorithms of QGrain

## Background

Grain size distribution (GSD) data have been widely used in Earth sciences, especially Quaternary Geology, due to its convenience and reliability. However, the usages of GSD are still oversimplified. The geological information contained in GSD is very abundant, but only some simplified proxies (e.g. mean grain size) are widely used. The most important reason is that GSD data are hard to interpret and visualize directly.

To overcome this, some researchers have developed the methods to unmix the mixed multi-modal GSD to some components to make the interpretation and visualization easier. These methods can be divided into two routes. One is end-member analysis (EMA) (Weltje, 1997) which takes a batch of samples for the calculation of the end-members. Another is called single-specimen unmixing (SSU) (Sun et al., 2002) which treats each sample as an individual.

The key difference between the two routes is that whether the end-members of a batch of samples are consistent. EMA believes that the end-members between different samples are consistent, the variations of GSD are only caused by the changing of fractions of the end-members. On the contrary, SSU has no assumption on the end-members, i.e. it admits that the end-members may vary between different samples.

Some mature tools (Paterson and Heslop, 2015; Dietze and Dietze, 2019) taking the EMA route have appeared, but there is no available public and easy-to-use tool for SSU. That the reason of creating QGrain.

## Fundamental

The math principle of SSU has been described by Sun et al. (2002).

In short, the distribution of a n-components mixed sample can be indicated as:

$y = f_1 * d_1(x) + ... + f_n * d_n(x)$,

where y is the mixed distribution, $f_i$ is the fraction of component i, $d_i$ is the base distribution function (e.g. Normal and Weibull) of component i, x is the classes of grain size.

The question is to get the distribution paramters of $d_i$.

Therefore, the unmixing problem can be coverted to an optimization problem:

minimize the error (e.g. sum of squared error) between $y_{test}$ and $y_{guess}$.

## Data preprocess

In fact the input data of each sample are two array. One is the classes of grain size, another is the distribution. Usually, there are many 0 values in the head and tail of distribution array. These 0 values were caused by the limit of test precision. In fact, they should be close to 0 but not equal to 0. This difference will bring a constant error which is large enough to effect the fitting result. QGrain will exclude these 0 values to obtain better performance.

## Local optimization

Due to the complexity of base distribution function, the error function is non-convex. At present, there is no high-efficiency method to find the global minimum of a non-convex function. So, an alternative solution is local optimization. Local optimization can converge to a minimum rapidly, but without guarantee that the minimum is global. Optimization problem also is a core topic of machine learning. Therefore, there are many mature local optimization algorithms that meet our requirement. Here we use Sequential Least SQuares Programming (SLSQP) (Kraft, 1988) algorithm to perform local optimization.

## Global optimization

With the increase of component number, the error function will become much more complex. It's difficult to get a satisfactory result if only use local optimization.

QGrain uses a global optimization algorithm called basinhopping (Wales & Doye, 1997) to improve the robustness.

This global optimization algorithm will not search the whole space but will shift to another initial point to start a new local optimization process after one local optimization process finished. That makes it has ability to escape some loacl minimum and keep the efficiency meanwhile.

## Base distribution function

At present, QGrain supports the following distribution types:

| Distribution Type | Parameter Number | Fitting Space | Skew |
|---|---|---|---|
| Normal[1] | 2 | Bin Numbers | No |
| Weibull | 2 | Bin Numbers | Yes |
| Gen. Weibull[2] | 3 | Bin Numbers | Yes |

1. Normal distribution againsts bin numbers is equal to Lognormal distribution againsts grain size (μm).
2. **Gen. Weibull** is General Weibull which has an additional location parameter.

## Steps of fitting

1. Data Loading
2. Get information (e.g. distribution type and component number)
3. Generate error function
4. Data preprocess
5. Global optimization (basinhopping)
6. Final optimization (another local optimization, SLSQP)
7. Generate fitting result by the parameters of error function

## Referances

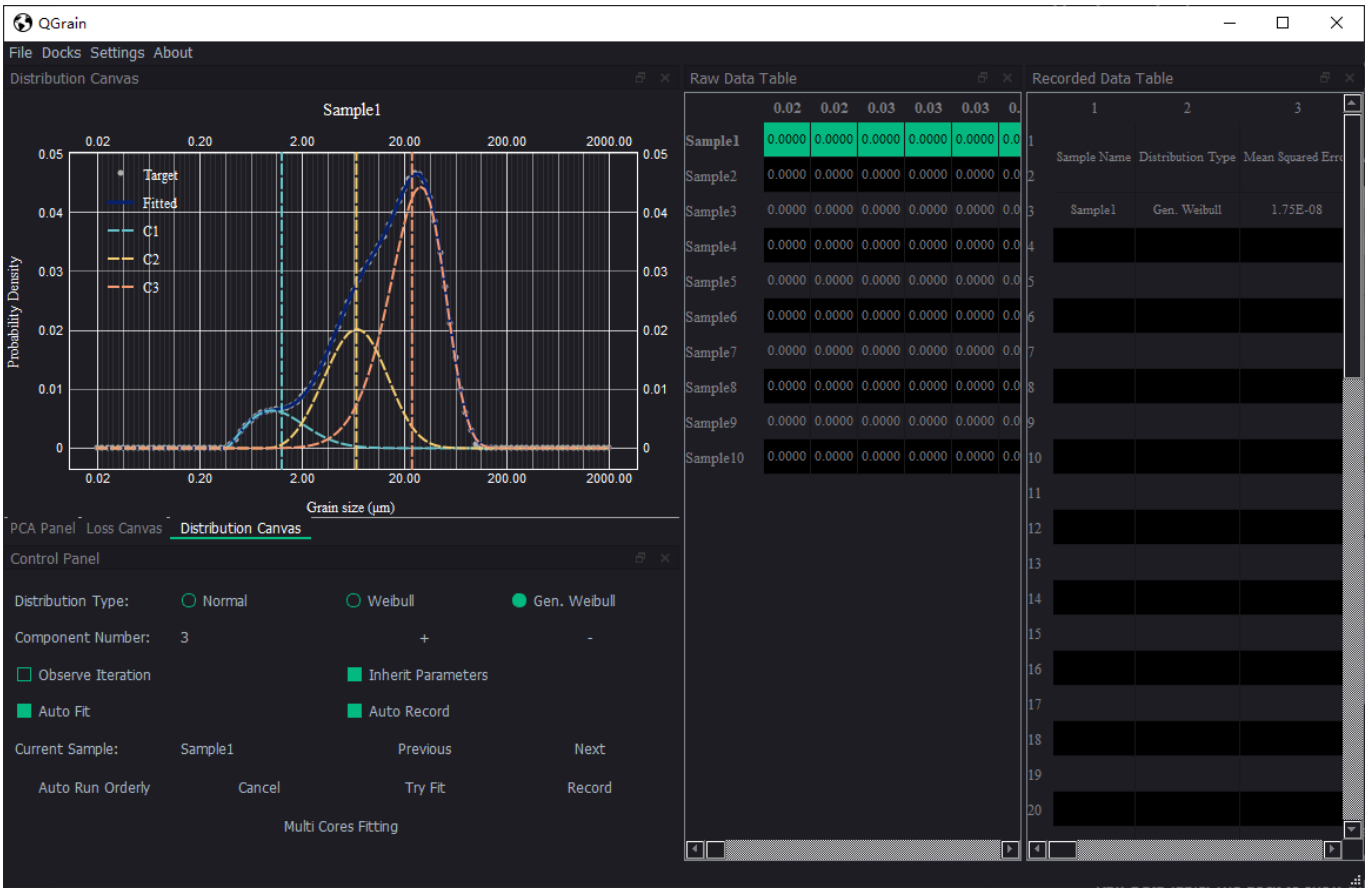

- Weltje, G.J. End-member modeling of compositional data: Numerical-statistical algorithms for solving the explicit mixing problem. Math Geol 29, 503–549 (1997) doi:10.1007/BF02775085

- Kraft, D. A software package for sequential quadratic programming. 1988. Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany.
- Wales, D J, and Doye J P K, Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. Journal of Physical Chemistry A, 1997, 101, 5111.

# How to fit the samples

By this, please make sure you have loaded the grain size distribution (GSD) data correctly.

If everything goes well, you will see the interface like this. By default, QGrain has fitted the first sample automatically.




If you are confused to some widgets, you can hover on it to see the tips.

## Workflow

The workflow of fitting samples is that:

1. Try fit one typic sample untill you are satisfied.

   You can adjust the component number and watch the chart of fitting result to find a proper value.

   If it can not return a correct result, you can check the **Ovserve Iteration** option to find the reason. The progress of fitting will be displayed by **Loss Canvas** and **Distribution Canvas**. Also, you can drag the lines to test whether the component number is proper.

   If it can return a proper result by giving the expected mean values, you can adjust the algorithm settings to refine the performance to let it can get the proper result automatically.

2. Test other samples with the component number.

3. If the component number are suitable for all samples, use auto fit to process them all.

4. If some results are not correct, cancel the fitting and return the step 1. If the incorrect results are not too many, you can fit and record manually.

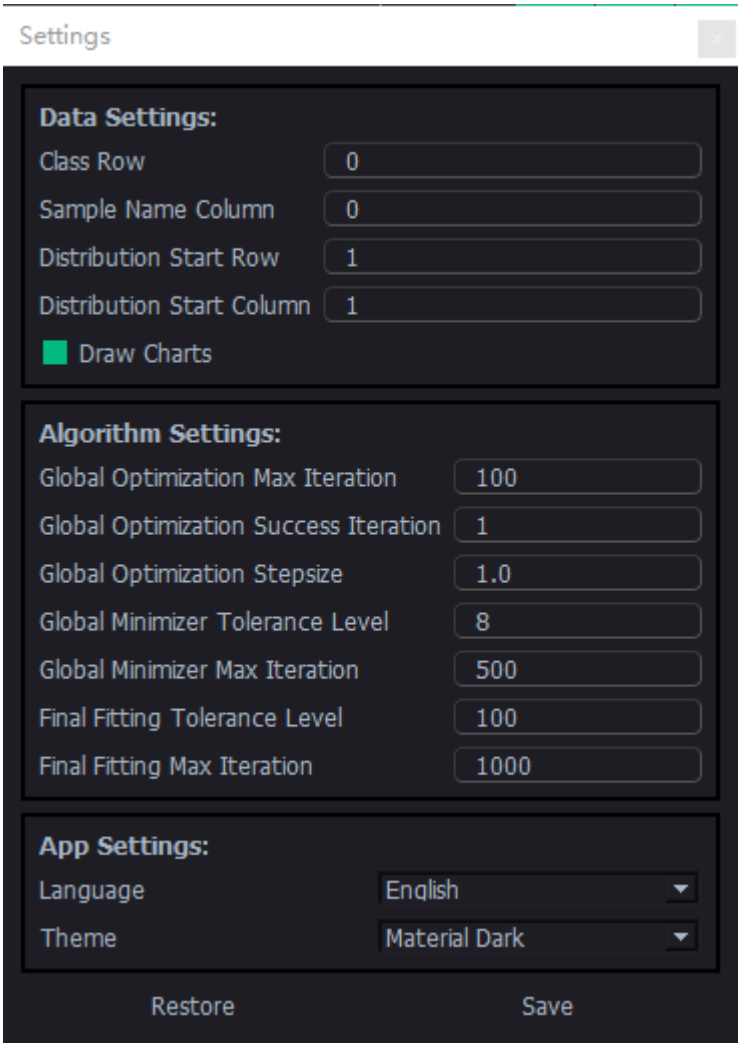
5. Click the **Save** option of **File** menu to save the fitting results to file.

# How to adjust algorithm

For some relatively simple samples (e.g. Loess), it's no need to adjust the algorithm settings. But for some fluvial and lacustrine deposits, the conditions of material sources and dynamics may be much more complex. In other words, the measured distribution may be the mixed result of many components (e.g. much than 4), and components may differ greatly. The complex distribution makes it's difficult to get a proper result. In order to deal with more complex situations, QGrain provides some algorithm settings.

If you are not familiar the algorithms of QGrain, click here for more information.

You can click the **Settings** menu to modify the settings of algorithm.




## Paramters

- **Global Optimization Max Iteration**: Max iteration number of global optimization. If the global optimization iteration has reached the max number, fitting process will stop.
- **Global Optimization Success Iteration**: It's one of the terminal conditions of global optimization. It means the iteration number of reaching the same minimum.
- **Global Optimization Stepsize**: The stepsize of searching global minimum. Greater stepsize will jump out the local minimum easier but may miss the global minimum.

- **Global Minimizer Tolerance Level**: The tolerance level of the minimizer of global optimization. Tolerance level means the accepted minimum variation (10 ^ -level) of the target function. It controls the precision and speed of fitting. It's recommended to use ralatively lower level in global optimization process but higher leverl in final fitting.
- **Global Minimizer Max Iteration**: Max iteration number of the minimizer of global optimization.
- **Final Fitting Tolerance Level**: The tolerance level of the minimizer of final fitting.
- **Final Fitting Max Iteration**: Max iteration number of the minimizer of final fitting.